6/prts

INSA1

ACCESSING DATABASES

This invention relates to a system and method for accessing data in a multi-user database environment and in particular relates to data access in a distributed database environment.

In the context of the present invention the term "database" relates to any collection of data and the term "distributed database" relates to any grouping of logically related databases distributed over a communications network. The communications network may comprise local area network (LAN), wide area network (WAN) or Internet components, for example. The term "distributed database management system" (DDBMS) relates to any software system that is associated with the management of a distributed database, and in particular one that makes the distribution appear transparent to a database user. In the context of the present invention the term "users" includes, for example, human operators, local computer systems and other computer systems. The term "data service" used herein concerns any database-related service that can be defined by one or more database operations such as data access functions. A data service manages and manipulates the data it has access to.

In a conventional distributed database data is distributed to a number of local databases which are connected to a communications network by respective database servers. Data services are provided by the local databases and their respective servers to client applications running on respective client terminals also connected to the communications network.

Large enterprise wide database systems are usually distributed on both a horizontal and a vertical fragmentation basis. In horizontally fragmented databases, data records of essentially the same type are distributed according to one or more related characteristics such as product type or place of manufacture. In vertically fragmented databases, data is distributed according to the relevant function of the data such as purchasing, sales and distribution. A consequence of database fragmentation is data replication.

There are significant drawbacks associated with data replication in database environments, namely:- a requirement for increased storage capacity within the database system; a requirement for multiple data updates; and the potential for data

inconsistency, for example. Despite these drawbacks fragmentation is common practice in enterprise wide systems, particularly where data is required on a regular basis to support a large number of processing tasks. In these systems data replication further provides built in redundancy which ensures continued database operation in the event of failure of a part of the communication network, or failure of a local database or its associated server. In addition data replication provides for more efficient local processing by reducing database response time, and contention for database resources and communication bandwidth.

In enterprise wide systems data is typically replicated to support local applications, that is to say, data is replicated to provide the type of data required to support a particular data service provided by a local database. In a heterogeneous environment the type of data replicated can vary significantly between local databases due to different local requirements. Data quality, as defined by various resource related quality parameters such as accuracy etc, can also vary significantly between local databases.

There are a number of factors that may affect the data quality of a local database. For instance, data updates may be made on a regular basis as determined by local database requirements or on a user initiated demand basis. Data updates may comprise unchecked but immediately up to date data or checked data that conforms to pre-determined quality standards such as accuracy and correctness. In addition, data updates may be delayed due to the non-availability of system resources such as network capacity or CPU time.

Data services provided by local databases in a distributed database system are therefore usually characterised by functional characteristics as defined by the data related operations the database supports, primarily determined by data structures and system architectures, and non-functional resource related characteristics such as data quality and system resources.

In known distributed databases the functional characteristics of the respective data services are presented to the client as published interfaces applications by the distributed database management system (DDBMS). Each published interface defines one or more database operations, that is data access functions, the data service can implement. When a client application selects an

interface the DDBMS selects an appropriate data service that can implement the database operations specified in the selected interface.

A major disadvantage of this type of database systems is that a published interface is usually associated with more than one data service. Thus, it is possible for any data service capable of implementing the database operations, or access functions, defined by the interface to be selected by the DDBMS. The non-functional quality characteristics of the data services such as data quality and system response time are hidden from the published interface, and hence selection of an appropriate data service by the DDBMS is independent of the non-functional data service requirements of the user. Thus, the non-functional characteristics of the selected data service only become apparent to the user once the data service has been implemented. In this regard, the data quality and response time of the data service is entirely dependent on an arbitrary selection of an available data service by the DDBMS.

Another disadvantage associated with known distributed database systems is that a popular data service can easily become over-subscribed, particularly when many client applications require concurrent real time access to the same data service. It is known to use resource management and resource allocation methods to control access to data services based on user-defined priority indications. However, this approach only affects the relative priority of a data access request in the database system.

According to a first aspect of the present invention there is provided a data management system comprising:-

a receiver for receiving data access requests for accessing data in a database system;

a register configured for storing an identifier for respective data services in the database system and, for each data service identified, first data relating to at least one respective data access function implemented by that data service and second data relating to data service resources relevant to at least one respective data access function;

a comparator for comparing a received data access request including at least a data access function requirement and a data service resource requirement with

respective first and second data to identify data services capable of accessing data in accordance with the request; and,

a selector for selecting a data service identified by the comparator for data access.

The system of the present invention is thus able to select a data service according to both a functional data access requirement and a non-functional resource-related quality of service requirement. In particular, the data management system readily allows a data access request to be matched to an available data service that has sufficient functionality and resources to implement the request. By considering both the functional and non-functional requirements the data management system is able to select a data service that is best suited to the requirements of the request. This provides for more discriminatory allocation and use of the data service resources provided in the database system. In particular, the data management system of the present invention can reduce over-subscription of highly functional and highly resourced data services. Data access requests which require lower levels of data access functionality or data service resources or both can be directed to less capable data services by the data management system.

Preferably, the register is further configured for storing third data for each data service identified, said third data relating to at least one data access tariff value relevant to at least one respective data access function. This allows different tariff values to be assigned to different data access functions implemented by the respective data services and additionally or alternatively to different database resources relevant to the data access functions.

Conveniently, the comparator is further configured for comparing a data access tariff requirement in said data access request with said third data. In this way the system of the present invention can further select an appropriate data service according to a tariff value requirement identified in the request. The use of different tariff values for different data services can further reduce the potential for data service over-subscription. In particular, a data service request can be used to balance data service functionality and resource requirements with tariff values applicable to those requirements. This can reduce the occurrence of over-specified data access requests and hence waste of database resources.

In preferred embodiments, the selector is configured to select a preferred data service according to a pre-determined selection strategy. In this way a common selection strategy can be implemented for each data access request.

Preferably, the selector is configured to select the data service having the lowest data access tariff value. Thus, when more than one data service is capable of implementing a data access request the most cost-effective data service is selected.

Conveniently, the system further comprises an event data recorder for recording event data relating to data service access events. This enables data access events to be analysed for dynamically altering the tariff values associated with the data services. In addition usage records can be maintained for each originating source of a data service request.

In preferred embodiments, the system further comprises a billing means for applying relevant data access tariff data to said event data for bill production. This readily provides for bill production.

Preferably, the system further comprises a connection manager for connecting users issuing data access requests to respective selected data services over a communications network. The connection manager provides a communication function in the data management system for controlling user access to selected data services. This guards against data corruption and unauthorised user access.

Conveniently, the connection manager comprises a monitor for monitoring the usage of the respective data services. This can be used to provide useful database usage information to the data management system.

In preferred embodiments, the connection manager further comprises access prevention means for limiting the number of users connected to each respective data service. In this way the connection manager can monitor the number of users connected to the respective data services and impose restrictions on further connections if a maximum number of connections is reached.

Preferably, the system further comprises an interface to the register for user access to data in the register. Thus, information in the register relevant to a data access request can be readily accessed and used to determine the precise form of the request. For example, first second and third data can be accessed to identify a data service that is capable of implementing a data access request in combination with a required resource related quality of service capability and associated tariff value.

Conveniently, the system further comprises an interface compiler for compiling data in the register for user access. This can prevent register data being corrupted by a user. Furthermore, the interface compiler enables easy and economical access to the data in the register.

In preferred embodiments, the system comprises part of a distributed database. Accordingly, the data management system can provide a centralised management function in a distributed database environment.

According to a second aspect of the present invention there is provided a data management system comprising:-

a receiver for receiving data access requests for accessing data in a database system;

a register configured for storing an identifier for respective data services in the database system and, for each data service identified, first data relating to at least one respective data access function implemented by that data service and second data relating to at least one data access tariff value relevant to at least one respective data access function;

a comparator for comparing a received data access request including at least a data access function requirement and a data access tariff requirement with respective first and second data to identify data services capable of accessing data in accordance with the request; and,

a selector for selecting a data service identified by the comparator for data access.

According to a third aspect of the present invention there is provided a method for managing access to data in a database system, the method comprising the steps of:-

storing in a register an identifier for data services provided in a database system and, for each data service identified, first data relating to at least one respective data access function implemented by that data service and second data relating to data service resources relevant to implementing at least one respective data access function;

receiving a data access request for accessing data in the database system, the request including at least a data access function requirement and a data service resource requirement;

comparing the received data access request with respective first and second data to identify data services capable of accessing data in accordance with the request; and,

selecting a data service identified by the comparison for data access.

Preferably, the method further comprises the step of storing third data for each data service identified, said third data relating to at least one data access tariff value relevant to at least one respective data access function.

Preferably, the method further comprises the step of providing user access to the data stored in the register.

Conveniently, the method further comprises the step of compiling the stored data for user access.

In preferred embodiments, the resource data comprises data from the group comprising:- data service response time, data accuracy, data correctness and time since last data update. Accordingly, data service resources can be defined in terms of data related parameters or system related parameters.

Preferably, the method is implemented in an object orientated software environment. This readily provides for integration of the system and method of the present invention in a legacy database system, that is to say a pre-existing system, and further provides for system scalability and software component re-use.

Conveniently, the step of storing data in the register comprises the step of publishing respective object orientated message interfaces using a communication protocol language.

The invention will now be described, by way of example only, with reference to the accompanying drawings, in which:-

Figure 1 shows a distributed database environment for a method and system of the present invention;

Figure 2 shows a functional block diagram of a distributed database including a distributed management system according to an embodiment of the present invention;

Figure 3 shows a detailed functional block diagram of part of a distributed database management system according to an embodiment of the invention; and,

Figure 4 is a flow diagram of a method embodying the present invention.

With reference to the system of Figure 1, a multi-user distributed database environment comprises a plurality of distributed nodes including database and client server nodes 10 connected together and to other client servers 12 over a communication network 14. The network 14 may comprise for instance a LAN, WAN, or the Internet depending on the extent of database distribution. A plurality of user access means defined by client terminals 16, which run client application code, are connected to each of the servers in a conventional client-server arrangement. The servers 10 are also connected to respective local databases 18 and together they provide data services to the client applications. The database of Figure 1 is an enterprise wide heterogeneous distributed database, comprising both hardware heterogeneity and system heterogeneity including differences in local database schemas, structures, data contents, query languages, interfaces and transaction protocols, for example.

The system of Figure 1 further comprises a distributed database management system (DDBMS) 20 as shown in Figure 2. The DDBMS 20 may be fully or partially replicated at each of the database server nodes 10 or at specific nodes only. The DDBMS architecture provides an object-oriented system that focuses on the provision of data services. The architecture arranges the DDBMS as a common system that acts as a broker separating data services 36 from client applications 34. With this architecture the communication network enables client applications to request data services from the DDBMS as if the data services were available locally, that is to say, the DDBMS renders the database distribution transparent to the clients 34.

Each data service publishes interface specifications that describes the functional data access operations the data service is capable of executing, and "protocol" specifications that describe the non-functional resource-related aspects of the data service. In this regard, a protocol specification defines values for one or more quality of service parameters, for example, values for data accuracy, data correctness, time since last update, data service response time etc. A protocol is a statement of the characteristics of an interface instance. A protocol does not define the characteristics of an interface, but defines the characteristics that will be exhibited by an interface when combined with the protocol. Each data service also publishes a "tariff" specification which describes the cost of a protocol and the number of client application connections the data service can accept for that tariff.

The tariff specification specifies the cost of executing a data access function in an interface while using a particular protocol. In the context of the present invention the term "tariff" may relate to a number of different costs and services, that is to say in the same way the term is used in the field of telecommunications to distinguish one set of costs from another, where the costs may be time and date dependent for example.

The DDBMS manages the publication of the data service interface, protocol and tariff specifications and subscription to the data services by the client applications.

When a client application requires access to a data service it identifies an interface that is capable of meeting its functional data access requirements and a protocol that is capable of meeting its non-functional quality of service requirements. If the interface is supported by more than one data service the client application will have a choice of data services. By identifying a protocol a client application enters into a pseudo contract with a data service for the provision of a data service that meets both its functional requirements and its resource related quality of service non-functional requirements. The terms of such a contract can be determined in a suitable service level agreement between the data service provider and the relevant database user. In the present invention, interface and protocol specifications may also be published by a client application in way manner analogous to an "invitation to tender" for a data service, or alternatively they may be determined by a standards body within an organisation. When making a so-called tender a data service publishes a tariff specification and builds an appropriate implementation. By publishing tariffs data services can use micro-economic systems to manage the loading on their respective underlying databases, that is to say access to the underlying databases can be controlled by changing the access costs of the respective data services in the respective tariff specifications.

The DDMBS separates the interface, implementation and quality of service issues in the provision of data services. Data services publish interface specifications, protocol specifications describing the non-functional characteristics of the data services and tariff specifications describing the cost of using the data services. Client applications subscribe to a data service by identifying an interface, and a protocol specification, and additionally or alternatively an associated tariff specification.

With reference now to Figure 3, the DDBMS comprises a data service broker 22 that connects client applications 34 to appropriate data services 36 via the communications network 14. The data service broker comprises a data service manager 24 and a data service register 26. The data service register comprises three related databases 28, 30 and 32. The databases 28, 30 and 32 store detailed information relating to data service functionality, quality and cost respectively for each data service in the distributed database environment. A data service register manager 27 manages the databases 28, 30 and 32.

The first database 28 stores information relating to the data access functions the respective data services implement. More specifically, each database server 10 publishes one or more interface specifications that specify the data access functions or operations the respective data services can implement. Interface specifications are stored in the database 28 for subsequent reference by the data service manager on behalf of client applications requiring access to particular data services. The name of each interface is registered in the register manager 27. The database 28 includes one entry for each interface comprising the name of the interface, the identity of the database server that submitted the interface, the identity of the data service or services the interface relates to and details relating to the interface definition including the interface definition source code.

In the present embodiment the functional interface specifications are defined using a standard communication language such as CORBA Interface Definition Language (IDL) (RTM). IDL is commonly used in object oriented systems. IDL allows software objects from different database systems and platforms to interact with one another.

The second database 30 stores information relating to the quality of service characteristics of each data service. More specifically, each database server publishes one or more protocol specifications that define quality of service characteristics of respective data services implemented by the database server. Protocols are stored in the database 30 for subsequent reference by the data service manager on behalf of client applications requiring access to particular data services. The name of each protocol is registered in the register manager 27. The database 30 includes one entry for each protocol comprising the name of the protocol, the identity of the database server that submitted the protocol, the identity of the data service or services the

protocol relates to and details relating to the protocol definition including the protocol definition source code. The database 30 also includes an entry for each resource related quality of service parameter defined in each respective protocol including the name of the protocol, the name of the data access function to which the parameter applies, the name of the parameter and the value of the parameter. In addition, the database 30 also includes an entry for each data access function relevant to a respective protocol including the name of the respective protocol, the name of the data access function and a ratio value defining the ratio of database queries or transactions allowed per connection session to a data service implementing the data access function. In the present embodiment the protocols are defined using a communication language based on Interface Definition Language (IDL), herein referred to as "Protocol Definition Language" (PDL).

The third database 32 stores information relating to the cost associated with each data service. More specifically, each database server publishes one or more tariff specifications which define the cost of using a data service and also the resource limitations imposed on the data service. Tariffs are stored in the database 32 for subsequent reference by the data service manager on behalf of client applications requiring access to particular data services. The name of each tariff is registered in the register manager 27. The database 32 includes one entry for each tariff comprising the name of the tariff, the identity of the database server that submitted the tariff, the identity of the data service the tariff relates to and details relating to the tariff definition including the tariff definition source code. The database 32 also includes an entry for each parameter defined in a tariff including the name of the tariff, the name of the data access function to which the parameter applies, the name of the parameter and the value of the parameter. In the present embodiment the tariffs are defined using a communication language based on Interface Definition Language (IDL), herein referred to as "Tariff Definition Language" (TDL). A tariff describes the cost and maximum usage of each data access function in an interface instance defined by a related protocol. Tariffs are defined separately from protocols so that data services can support the same protocol at a variety of different costs depending on their respective workload and data access implementation techniques.

The data service manager is connected to the register for identifying data services listed in the register that are capable of meeting the requirements of a user

defined data service request from a client application. Each client application is provided with a data request means 44 for issuing a data access request to the data service broker. Data service requests are received from client applications at a communications interface 46 which is connected to a network connection manager 48. The connection manager connects client applications to appropriate data services selected by the data service manager. The data service manager comprises an IDL compiler 38, a PDL compiler 40 and a TDL compiler 42 for querying the respective databases 28, 30 and 32 in the register. A comparator 50 is provided in the data service manager for comparing received data access requests from client applications with the interface, protocol and tariff specifications in the respective databases 28, 30 and 32. The comparator identifies the data services that are capable of meeting the requirements of the request.

In an alternative embodiment (not shown) the compilers 38, 40 and 42 are provided at the client applications to allow the client applications to access the interface, protocol and tariff specifications in the databases 28, 30 and 32.

The data service manager is further provided with a data service selector 52 which is programmed to select the most appropriate data service identified by the comparator 50. The selector is programmed to select a data service according to pre-determined selection criteria based on a data access cost value defined in the respective tariff specifications.

The connection manager comprises a monitor 52 which is connected to a monitoring database 54. The monitor 52 monitors the usage of all the data services listed in the register. The monitor also monitors the current workload of each database server and maintains a record in the database 54 of all connections made to data services by the client applications. The monitoring database 54 comprises one entry for each data service that has registered one or more tariffs with the register. The monitoring database records the name and address of each respective data service and is provided with a counter 56 which maintains a count of the current usage of each of the data services by the client applications. This information is used by the network connection manager to determine whether a preferred data service is available, that is to say whether the preferred data service or its underlying database server can accept further connections. If a pre-determined maximum number of

connections to a data service or a database server is reached subsequent connections are prevented by an access prevention means 58 in the connection manager.

The data service broker is further connected to a billing system 60 for bill production. The billing system determines the cost associated with each data service connection using tariff data provided by the data service register and event data provided by the monitoring database. The billing system bills subscriber accounts for use of the data services provided by the distributed database.

The monitoring database 54 is further associated with a cost determining means 62 that adjusts the tariff cost data of the respective data services in the database 32 according to the current usage of the respective data services or total usage over a measured period.

With reference now to Figure 4, the flowchart represents a method of managing access to data in a multi-user distributed database environment according to an embodiment of the invention. In this embodiment the method is implemented in the distributed database system described with reference to Figures 1, 2 and 3.

In the first step 100 a client application that requires access to data issues a remote call for connection to the data service broker over the communication network 14. Once the connection is established the client application is asked by the DDBMS to provide a data service request in step 102. Either prior or subsequent to this step the client application may prompt the user of the system for details of a data service request. For example, the client may present a graphic user interface (not shown) having drop down menus of the interface, protocol and tariff names for selection by the user to define the required data access request.

A fully defined data service request includes the name of an interface, a protocol and a tariff, however only the name of an interface and a protocol or a tariff is required in this present embodiment. In step 104 the client application asks the user if access to the interface, protocol and tariff databases is required, that is for access to the respective interface, protocol and tariff parameters etc contained in the databases. If access to one or more of the databases is required the data service manager provides access in step 106. The client application queries the or each database in step 108 to obtain information relating to the respective published interface, protocol and tariff specifications. In step 110 the client application issues a data access request to the data service broker. The data access request is defined by

the client application using information obtained in step 108 or from knowledge derived from previous database usage.

The data service manger determines whether the data access request is fully defined in step 112, that is to say whether the request includes the name of a tariff since the respective tariff specification identifies both a related interface and protocol. If the data access request does not include a tariff name the data service manager determines whether the request includes the name of a protocol in step 114. If the data service request fails to provide the name of at least a protocol the request is terminated and the method returns to step 102.

If the request is fully defined the tariff database is searched in step 118 by the data service manager for data services that exactly meet the requirements of the data access request. The data service manager identifies the name and address of one or more appropriate data services that meet the requirements of the request in step 120. In this respect it should be noted that if a tariff uniquely identifies a particular data service the name and address of only that data service will be identified. In step 122 the data service manager determines whether more than one data service is capable of meeting the requirements of the request. If so the names of the data services are returned to the client application in step 124 and the client application is provided with the option of accessing information relating to the respective data services in the interface, protocol and tariff databases in step 126. If the client application accesses one or more of the databases in step 127 the procedure follows that of steps 106 and 108. If the client application identifies a preferred data service in step 128 the data service manager selects that data service on behalf of the client application in step 129. If no preferred data service is identified any one of the appropriate data services is selected in step 129a.

If the data access request is only part defined in the sense that it identifies a protocol but not a tariff the protocol database is searched in step 130 by the data service manager for data services that meet the requirements of the partly defined request. The data service manager identifies the name and address of one or more appropriate data services that meet the requirements of the request in step 132. In this respect it should be noted that if a protocol uniquely identifies a particular data service the data service manager will return the name and address of only one data service. In step 134 the data service manager determines whether more than one

data service is capable of meeting the requirements of the request. If so the names of the data services are returned to the client application in step 136 and the client application is provided with the option of accessing information relating to the respective data services in the interface, protocol and tariff databases in step 138. If the client application accesses one or more of the databases in step 139 the procedure follows that of steps 106 and 108. If the client application identifies a preferred data service in step 140 the selector selects that data service on behalf of the client application in step 141. If no preferred data service is identified by the client application the data service manager accesses the tariff database in step 142 and queries the cost parameters in each of the respective tariff specifications and selects the data service having the lowest usage cost tariff on behalf of the client application in step 143.

Once a data service has been selected the connection manager accesses the monitoring database and queries the counter to determine the current usage of the selected data service and that of the respective database server in step 158. If the current usage is below a maximum value the connection manager calls the data service address and establishes a connection between the client application and the respective database server in step 160. If the usage exceeds a pre-determined value the client application is informed in step 159 and the method returns to either step 124 or step 136 depending on the original data access request.

Access to the selected data service is provided to the client application by the broker in step 162. The connection manager accesses the monitoring database in step 164 and increments the respective data service usage counter by the value of the usage parameter specified in the respective tariff specification of the selected data service. Once the connection ends in step 166 the usage counter decrements by the value of the respective usage parameter in step 168.

Pseudo code for an embodiment of the present invention implemented in an object-oriented environment will now be described.

Interfaces, protocols and tariffs are instantiated as objects by their originator and submitted to the data service broker using publishX() methods, where X is either:- Interface, Protocol or Tariff. For example, the method publishTariff (t, DS) publishes a tariff t for a data service DS. Interfaces and protocols are published by data services or client applications. A data service publishes:- i) an interface when it

can provide a data service that can support database operations defined in the interface; ii) a protocol when it can provide a data service having a defined quality of service; and, iii) a tariff when it supports an interface and protocol in a data service. A client application publishes an interface and protocol when it requires a data service that precisely meets its respective data and resource related quality of service requirements. A published interface or protocol may be used by any data service to register itself with the data service broker or by any client application in a data access request. In this respect a data service can become a registered supplier of a particular data service by publishing a respective interface, protocol and tariff.

The following discussion relates to an enterprise wide customer database implemented by a telecommunications company.

A known interface for retrieving customer data in JAVA (RTM) programming language is:-

```
interface customer_call_data {
        public CustomerDetails getCustomerDetails (int cust_id);
}
```

This interface specification states that any object implementing the interface customer_call_data will provide a method getCustomerDetails() that returns details of a customer given a customer's identifying number. The interface hides the implementation of the getCustomerDetails() method by only describing the format of the request and not the method or code for carrying out the request. The interface does not provide any information relating to the quality of service of the method getCustomerDetails().

An example of an interface implemented by a data service for the interface customer_call_data is as follows:-

```
class customers implements customer_call_data
        public CustomerDetails getCustomerDetails (int cust_id) {
                ..... algorithms for method
        }
}
```

The class customers implements the interface customer_call_data as it includes an implementation of the getCustomerDetails method. The implementation can be any set of code that retrieves customer records. For example, the algorithms may relate to a sequential search through the data, an index search based on a customer identifying number or a search request to a system operator. Although the implementation is hidden from the client application by the interface, the performance of the data service implementing the interface is not. In the system and method of the present invention the performance of a data service is made explicit by defining a protocol

An example of a protocol for the customer_call_data interface is:-

```
protocol summary_data describes customer_call_data {
        CustomerDetails getCustomerDetails (int cust_id) {
                accuracy => 0.9
                min_execution_time => 100
                max_execution_time => 1000
                timeliness => 24;
        }
}
```

The protocol specification states four properties of the getCustomerDetail() method namely:- i) the accuracy of the data returned by the method will be greater than or equal to 90%, ii) the minimum execution time will be 100 units, iii) the maximum execution time will be 1000 units, and iv) the data will be updated at least every 24 hours.

The parameters specified in a protocol depend on the nature of the data service to which the protocol relates. Each data service monitors its performance and determines values for each of the parameters listed in the protocols to determine whether the data service can support the performance requirements set out in the protocols it is associated with. Parameter values are monitored on a periodic basis and communicated to the broker from the data services to update the register databases. Once determined the updated parameter values can be input at the data

service for transmission to the data service manager by an operator or alternatively the values may be generated automatically using a performance monitor algorithm embedded in software installed at the data service.

An interface may have many protocols that meet different data service requirements. For example, an alternative protocol for the customer_call_data interface is:-

```
protocol real_time_summary_data describes customer_call_data {
        CustomerDetails getCustomerDetails (int cust_id) {
                accuracy => 0.5
                min_execution_time => 10000
                max_execution_time => 50000
                timeliness => 0;
        }
}
```

This protocol specification can be used with the same interface to provide a data service having a different quality of service. The protocol provides immediately up to date data (timeliness = 0), but the execution time is slower because the data service has to compete with other data services for example, and the accuracy is lower because the data has not been pre-processed to identify errors. In this way a client application can obtain immediately up to date but less accurate customer data by using a data service that implements the same customer_call_data interface with the real_time_summary_data protocol.

In addition, the protocol specification can be used to define relationships between methods. For example, another protocol for the customer_call_data interface is:-

```
protocol summary_data2 describes customer call data {
        void Connect() {
                min_execution_time => 10
                max_execution_time => 50
        }
```

```
        CustomerDetails getCustomerDetails (int cust_id) {
                accuracy ≥ 0.5
                min_execution_time =>10000
                max_execution_time => 50000
                timeliness = 0;
        }
        void Disconnect() {
                min_execution_time => 10
                max_execution_time => 50
        }
        ratio Connect:1, getCustomerDetails:100, Disconnect:1
}
```

This protocol specification describes the relationship between calls to the methods Connect(), getCustomerDetails() and Disconnect() as one call to Connect() and Disconnect() for every 100 calls to getCustomerDetails(). This relationship can be used to prevent excessive use of the underlying databases by a client application. Moreover, a client application can use this information to select the most appropriate protocol for its requirements.

The following is an example of a tariff for the protocol summary_data:-

```
tariff summary_data_cost on summary_data {
        CustomerDetails getCustomerDetails (int cust_id) {
                resource_usage => 10
                cost_per_execution => 200
        }
}
```

In this example the tariff summary_data_cost describes the cost of using the method getCustomerDetails() in the interface customer_call_data. The method costs 200 units for each call and the execution cost of the method in terms of data service resource usage is 10 units. The data service broker in combination with the billing system uses the information in the tariff definition to charge client applications for

using a data service and also to select the lowest cost data service that can meet a client's requirements. The resource usage and cost per execution parameters are defined separately as some data services may not impose large workloads on the underlying database systems but their use may be costly for other reasons, for example because a significant amount of pre-processing is required to achieve the accuracy specified in the respective protocol.

The resource_usage parameter is used by the connection manager to monitor the number of client applications using a data service. The resource_usage parameter allows the connection manager to determine the number of client application connections a data service can sustain.

For example, a tariff for the protocol summary_data2 is:-

```
tariff summary_data2_cost on summary_data2 {
        void Connect {
                resource_usage => 2
                cost_per_execution => 10
        }
        CustomerDetails getCustomerDetails (int cust_id) {
                resource_usage => 10
                cost_per_execution => 200
        }
        void Disconnect() {
                resource_usage => 2
                cost_per_execution => 10
        }
}
```

If each client application connection can execute one method at a time, the maximum workload per connection is 10 units. In this respect if the maximum workload of the data service is 100 units it is capable of supporting 10 client application connections.

A data service that supports a tariff must implement the respective interface associated with the tariff and select an implementation strategy that will support the respective protocol.

When a data service is required by a client application the client application connects to the data service broker using a Remote Method Invocation (RMI) call lookup method. Once connected the client application issues a user defined data access request to the data service broker. The data access request identifies either an interface, a protocol and a tariff, an interface and a protocol or an interface only if that interface is only associated with one particular protocol. If a tariff relates to only one protocol, a data service request that specifies a tariff also identifies the relevant interface and protocol associated with the tariff. In this type of request the client application identifies all the properties of the data service required, that is to say the functional and quality of service characteristics as well as the access usage cost. If on the other hand a tariff is not specified in the data service request the data service manager uses a pre-determined strategy based on lowest usage cost to select an appropriate data service for connection to the requesting client application.

The data service manager implements the following programming interfaces for selecting an appropriate data service including:- i) a broker_access interface for locating one or more appropriate data services; ii) a protocol_definition interface for querying the or each available protocol; and, iii) a tariff definition interface for querying the or each available tariff.

An example of the broker_access interface is:-

```
interface broker_access
        String []findDSInterface( String interface_name );
        String []findDSProtocol( String protocol_name );
        String []findDSTariff( String tariff_name );
        String getDSInterface( String service_addr );
        String getDSProtocol( String service_addr );
        String getDSTariff( String service_addr );
```

A client locates an appropriate data service using find data service methods, for example:-

```
String [] service = broker.findDSTariff( tariff_name );
```

Each findDSType call returns one or more available data service addresses. A data service address is a standard RMI connection address. If more than one data service is available the client application can query the register's databases. The connection address of an identified data service is passed to the getDSType methods of the broker_access interface to obtain the name of the interface, protocol and tariff of the data service. The client application can query the data service register about the properties of a named protocol or tariff using the respective protocol_definition and tariff _definition interfaces. These interfaces are implemented by the data service manager to provide access to the register's databases.

An example of a protocol definition and a tariff definition interface is:-

```
interface protocol_definition {
        String getProtocolNames();
        String getProtocolMethods( String protocol_name );
        String getProtocolParameters( String protocol_name,
                                String method_name );
        Object getProtocolParameterValue( String protocol_name, String
                                method_name, String param_name );
        int [] getProtocolRatio( String protocol_name );
}

interface tariff_definition {
        String getTariffNames();
        String getTariffMethods( String tariff_name );
        String getTariffParameters( String tariff_name,
                                String method_name );
        Object getTariffParameterValue( String tariff_name,
                                String  method_name, String param_name );
}
```

The workload of the selected data service is determined when the client application requests connection. The connection manager estimates the potential workload that the client application will impose on the data service and either rejects or accepts the request. This is done using the resource_usage parameter for the tariff being used. If the connection is rejected the client application must select another one of the data services from the list returned by the findDStype method. If the connection is accepted the data connection manager connects the client application to the data service. The address returned by the findDSType method is used to open the connection using an RMI call. A data access class object is instantiated by the TDL compiler when the connection manager requests the connection. The data access class acts as an object wrapper around the data service. The data access object implements the interface but not the functions of the data service. The data access object records the implementation of the selected interface in the monitoring database and passes the RMI call to the data service. When the data service completes the execution of the method the result is passed back to the client application through the data access class object wrapper. The connection manager assesses the workload of each data service using the monitoring database. Each data access object increments or decrements the counter in the monitoring database for each method call to a data service. When a data access object is instantiated it creates a data service event record in the monitoring database that records details relating to the connection and the cost of executing each method for example, and further initialises access counters for each method in the data service.

For example, the tariff summary_data_cost is compiled to give the data access class:-

```
class DA_summary_data_cost {
        MonitoringDB m;
        Object data_service;

        public DA_summary_data_cost() {
                m = ...establish database connection...;
                data_service = ...establish data service connection...;
        }
```

```
public CustomerDetails getCustomerDetails( int cust_id ) {
    ...increment counts in m...
    CustomerDetails r = data_service.getCustomerDetails
                                                ( cust_id );
...decrement count in m...
    return r;
}
}
```

In this way when a method in the data access object is called it increments a counter for the relevant data service in the monitoring database by the resource_usage parameter of the data access object's tariff. When the method call is finished the data access object decrements the counter by the same amount. The monitoring database thus contains the current usage of the data service.

In summary therefore, it will be seen that by separating the protocols and interfaces that are used to access a data service it is possible to use different database implementation techniques to support different types of data manipulation. The protocols supported by a data service determine the type of data structures that are implemented by the data service, for example, the data replication and distribution strategy of the underlying database. A protocol does not determine the type of interface used by the client and does not provide the client with explicit knowledge of the underlying database management system that supports the data service. Clients select an appropriate protocol for the type of data manipulation they wish to perform and data services undertake to provide a certain set of performance characteristics to clients using a protocol.

Tariffs are used to "charge" a client for the use of a data service and are based on the type of data provided by the service, the protocol and interface adopted by the client and the use made of the data by the client. For example, a data service may provide a transaction protocol optimised for real time transaction processing involving simple database searches and an analytical protocol optimised for more complex data analysis. The transaction protocol may be implemented using a simple relational database that is relatively easy to maintains whereas the analytical protocol may require complex data structures such as a data warehouse that are more difficult

manage. Hence, tariffs associated with the transaction protocol will be low compared with those associated with the analytical protocol. For instance, a client that connects to the data service using the transaction protocol and an interface that performs simple record searches will have a low tariff and good performance. However, a client that connects to the data service using the same transaction protocol but with an interface optimised complex data analysis will have the same low tariff but will experience poor performance. On the other hand a client that wishes to perform complex data analysis should select the analytical protocol. The analytical protocol will provide good performance, but because the data structures are more complex and difficult to manage the tariff will be higher. In this respect, tariffs allow the data service to charge the clients based on the complexity of supporting a service and managing the data.

It will be understood that the invention is not limited to the particular embodiments described in the above description, but includes alternative embodiments that are readily apparent to those skilled in the art. For example, the database could be provided by a single autonomous database rather than a distributed database. Moreover, the invention does not have to be implemented in an object-oriented software environment.